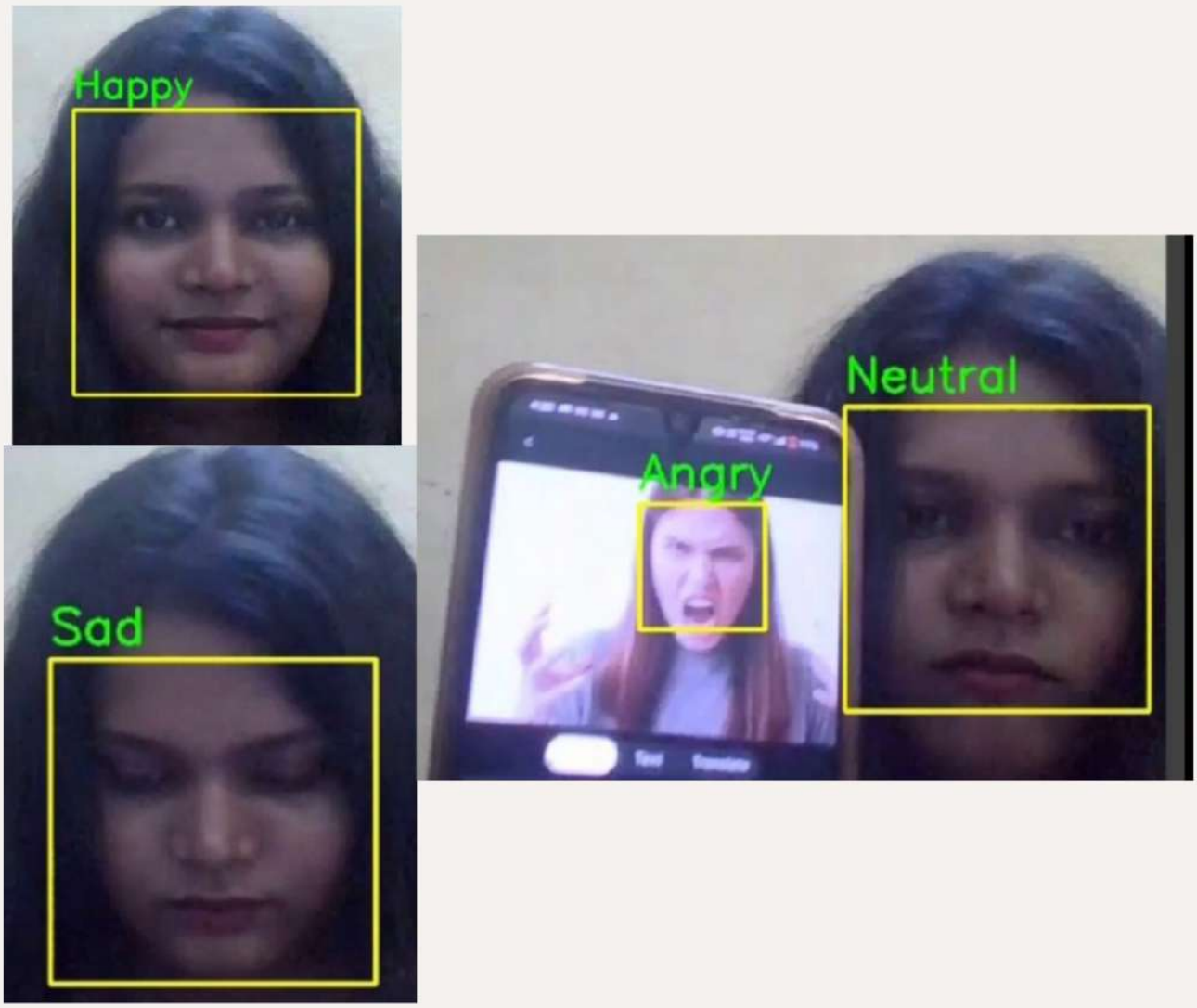
Face Emotion Detection



In this project our system can understand the human emotions by theri expression

Steps

1. Import libraries and datasets

2. We have different types of photos collection (fear, angry etc) so we have to categories them.

3.Then we have to train and test the dataset

n [1]:
```
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping
import numpy as np
import os
import cv2
from tensorflow.keras.layers import  BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
```

n [2]:
```
INIT_LR = 1e-4
EPOCHS =8
BS = 32
DATASET ="E:\3.DeepLearningProject\train"
DATASET
```

ut[2]:
```
'E:\x03.DeepLearningProject\train'
```

n [3]:
```
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri
NUM_CLASSES = len(CATEGORIES)
IMG_SIZE = 48
```

n [4]:
```
import os
import cv2
import matplotlib.pyplot as plt

# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the

# Select the "angry" category
emotion_category = "angry"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 2
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
```

```python
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)

    # Convert BGR image to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.title(emotion_category)
    plt.axis("off")  # Turn off axis labels
    plt.show()
```

In [5]:

```python
import os
import cv2
import matplotlib.pyplot as plt

# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the

# Select the "angry" category
emotion_category = "disgust"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 3
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)
```

In [6]:

```python
import os
import cv2
import matplotlib.pyplot as plt

# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surprise"]

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the string

# Select the "angry" category
emotion_category = "fear"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 3
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)

    # Convert BGR image to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.title(emotion_category)
    plt.axis("off")  # Turn off axis labels
    plt.show()
```

fear

In [7]:
```
# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the

# Select the "angry" category
emotion_category = "happy"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 3
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)

    # Convert BGR image to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

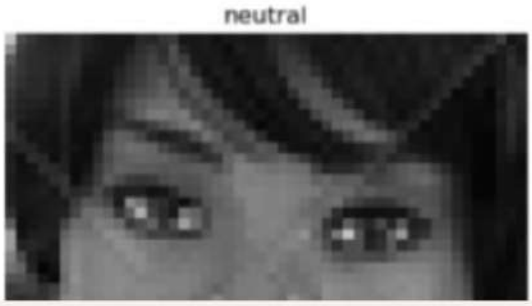
In [7]:
```python
# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the

# Select the "angry" category
emotion_category = "happy"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 3
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)

    # Convert BGR image to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```python
    plt.imshow(image_rgb)
    plt.title(emotion_category)
    plt.axis("off")  # Turn off axis labels
    plt.show()
```

happy

In [8]:

```
# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surprise"]

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the string

# Select the "angry" category
emotion_category = "neutral"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 3
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)

    # Convert BGR image to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.title(emotion_category)
    plt.axis("off")  # Turn off axis labels
    plt.show()
```



In [9]:
```python
# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the

# Select the "angry" category
emotion_category = "sad"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 3
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)

    # Convert BGR image to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    plt.imshow(image_rgb)
    plt.title(emotion_category)
    plt.axis("off")  # Turn off axis labels
    plt.show()
```

In [10]:
```python
# Define the list of emotion categories
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri

# Set the path to the directory containing the images
dataset_path = r"E:\3.DeepLearningProject\train"  # Note the 'r' before the

# Select the "angry" category
emotion_category = "surprise"

# Create the path to the specific category directory
category_path = os.path.join(dataset_path, emotion_category)

# Get the list of image filenames in the category directory
image_filenames = os.listdir(category_path)

# Select two or three images from the list
num_images_to_display = 3
selected_image_filenames = image_filenames[:num_images_to_display]

# Loop through the selected image filenames
for image_filename in selected_image_filenames:
    # Create the full path to the image file
    image_path = os.path.join(category_path, image_filename)

    # Load the image using OpenCV
    image = cv2.imread(image_path)

    # Convert BGR image to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```python
    plt.imshow(image_rgb)
    plt.title(emotion_category)
    plt.axis("off")  # Turn off axis labels
    plt.show()
```

surprise

In [16]:
```
DATASET = r'E:\3.DeepLearningProject\train'  # Use raw string literal
CATEGORIES = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surpri
NUM_CLASSES = len(CATEGORIES)
IMG_SIZE = 48
def load_dataset():
    data = []
    labels = []
    for category in CATEGORIES:
        path = os.path.join(DATASET, category)
    for img_name in os.listdir(path):
        img_path = os.path.join(path, img_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        data.append(img)
        labels.append(CATEGORIES.index(category))

    data = np.array(data, dtype="float32") / 255.0
    labels = np.array(labels)
    return data, labels

data, labels = load_dataset()
labels = to_categorical(labels, NUM_CLASSES)
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=
```

In [17]:
```
datagen = ImageDataGenerator(
    rescale=1.0 / 255.0,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
```

```
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)
```

In [18]:
```
train_generator = datagen.flow_from_directory(
    DATASET,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BS,
    class_mode='categorical',
    color_mode='grayscale',
    shuffle=True
)
```

```
Found 28709 images belonging to 7 classes.
```

In [19]:
```
from tensorflow.keras.layers import Input, GlobalAveragePooling2D, Dense, Dr
from tensorflow.keras.regularizers import l2
baseModel = MobileNetV2(weights=None, include_top=False, input_shape=(IMG_SI
headModel = baseModel.output
headModel = GlobalAveragePooling2D()(headModel)
headModel = Dense(256, activation="relu", kernel_regularizer=l2(0.01))(headM
headModel = BatchNormalization()(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(128, activation="relu", kernel_regularizer=l2(0.01))(headM
headModel = BatchNormalization()(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(NUM_CLASSES, activation="softmax")(headModel)

model = Model(inputs=baseModel.input, outputs=headModel)
opt = Adam(learning_rate=INIT_LR)
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accu
```

In [20]:
```
early_stopping = EarlyStopping(monitor="val_loss", patience=3, restore_best_

# Train the model
history = model.fit(
    train_generator,
    epochs=EPOCHS,
    batch_size=BS,
    callbacks=[early_stopping]
)
```

```
Epoch 1/8
898/898 [==============================] - ETA: 0s - loss: 8.3014 - accuracy:
0.1606WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 1427s 2s/step - loss: 8.3014 - acc
uracy: 0.1606
Epoch 2/8
898/898 [==============================] - ETA: 0s - loss: 6.9486 - accuracy:
0.1876WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 802s 892ms/step - loss: 6.9486 - a
ccuracy: 0.1876
Epoch 3/8
898/898 [==============================] - ETA: 0s - loss: 5.7639 - accuracy:
0.1989WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 782s 871ms/step - loss: 5.7639 - a
ccuracy: 0.1989
Epoch 4/8
898/898 [==============================] - ETA: 0s - loss: 4.6589 - accuracy:
0.2137WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 785s 874ms/step - loss: 4.6589 - a
ccuracy: 0.2137
Epoch 5/8
898/898 [==============================] - ETA: 0s - loss: 3.7595 - accuracy:
0.2253WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 790s 879ms/step - loss: 3.7595 - a
ccuracy: 0.2253
Epoch 6/8
898/898 [==============================] - ETA: 0s - loss: 3.0909 - accuracy:
0.2313WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 791s 881ms/step - loss: 3.0909 - a
ccuracy: 0.2313
Epoch 7/8
898/898 [==============================] - ETA: 0s - loss: 2.6544 - accuracy:
0.2417WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 779s 867ms/step - loss: 2.6544 - a
ccuracy: 0.2417
Epoch 8/8
898/898 [==============================] - ETA: 0s - loss: 2.3940 - accuracy:
0.2448WARNING:tensorflow:Early stopping conditioned on metric `val_loss` whic
h is not available. Available metrics are: loss,accuracy
898/898 [==============================] - 713s 794ms/step - loss: 2.3940 - a
ccuracy: 0.2448
```

In [24]:

```
from sklearn.metrics import classification_report

print("[INFO] evaluating network...")
predIdxs = model.predict(X_test, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
```

```
[INFO] evaluating network...
20/20 [==============================] - 3s 146ms/step
```

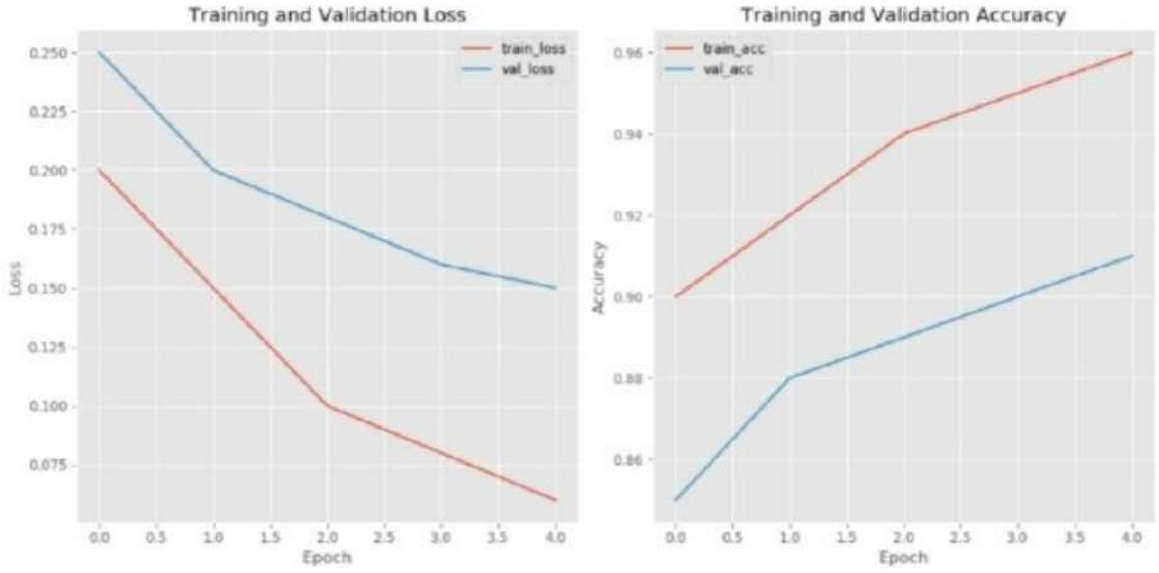
In [27]:
```
from sklearn.preprocessing import LabelEncoder
```

In [29]:
```
label_encoder = LabelEncoder()
y_test_int = label_encoder.fit_transform(np.argmax(y_test, axis=1))
y_pred = model.predict(X_test)
y_pred_int = np.argmax(y_pred, axis=1)

print(classification_report(y_test_int, y_pred_int, labels=np.arange(NUM_CLA
```

```
20/20 [==============================] - 4s 222ms/step
D:\ANACONDA\ANA\envs\aiworks\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use `zero_division` par
ameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\ANACONDA\ANA\envs\aiworks\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

       angry       0.00      0.00      0.00     635.0
     disgust       0.00      0.00      0.00       0.0
        fear       0.00      0.00      0.00       0.0
       happy       0.00      0.00      0.00       0.0
     neutral       0.00      0.00      0.00       0.0
         sad       0.00      0.00      0.00       0.0
    surprise       0.00      0.00      0.00       0.0

   micro avg       0.00      0.00      0.00     635.0
   macro avg       0.00      0.00      0.00     635.0
weighted avg       0.00      0.00      0.00     635.0
```

In [29]:
```
model.save("face_emotion_recognition_model.h5")
```

In [38]:
```
plt.style.use("ggplot")
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(history["loss"], label="train_loss")
plt.plot(history["val_loss"], label="val_loss")
plt.title("Training and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history["accuracy"], label="train_acc")
plt.plot(history["val_accuracy"], label="val_acc")
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()

plt.tight_layout()
plt.show()
```



In [41]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
# Generate the confusion matrix
cm = confusion_matrix(y_test.argmax(axis=1), predIdxs)
plt.figure(figsize=(8, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=CATEGORIES, yticklabels=CATEGORIES)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```